Escuela Politécnica Superior

Pruebas sobre sitios web

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Héctor Sansano Miralles

Tutor/es: Jaume Aragonés Ferrero

Septiembre 2017

Universitat d'Alacant
Universidad de Alicante

1 Justificación y objetivos

En el mundo de la informática y la tecnología los sitios web son una parte importante, su nacimiento supuso una revolución, primero en la comunidad científica y académica, más tarde, con su popularización en los entornos más domésticos.

Existen otras tecnologías actualmente emergentes que están aumentando el terreno a los sitios web

La aparición del teléfono inteligente ha abierto nicho de mercado. Las aplicaciones móviles también han quitado espacio a los sitios web. Sin embargo aún existe gran parte debido a sus ventajas como la de ser "multiplataforma" y así tener que adaptarse a unos estándares.

A día de hoy los sitios web son una parte importante de la informática. Su desarrollo comprende diferentes partes de esta como: servidores, sistemas, programación backend, programación frontend, jerarquía de objetos, diseño gráfico etc… Básicamente un sitio web se compone de esto.

Los objetivos principales de este trabajo estudiar y profundizar en el funcionamiento de los sitios web. Analizar el contenido del sitio, sus principales elementos estructurales y elementos del HTML, principalmente sus elementos un poco más complejos como los enlaces a otras páginas de mismo sitio web o información enviada en sus formularios. Se intentará probar estos elementos para medir el rendimiento del sitio e intentar encontrar algunos errores en el diseño de este. Para ello la idea principal es crear un software que a partir de una url haga un análisis de este.

Índice de contenidos

2 Introducción

La navegación por la web se ha vuelto un importante con el tiempo. La red de internet nació con ARPANET con el objetivo de intercambiar información entre entidades académicas y estatales de los Estados Unidos. Se puede decir que la red de internet es usada para telefonía, radio, televisión. Nuestro trabajo estará enfocado a la parte del world wide web o la navegación web.

La navegación web poco se ha ido popularizando poco a poco. La mejora de las redes de interconexión y la oferta de los operadores de acceso a internet ha hecho que crezca el número de sitios web en internet. A la publicidad en la televisión, radio o prensa se le sumó el medio de internet. Empresas comenzaron a abrir sus sitios web oficiales donde se publicitaban y mostraban sus servicios.

Por eso actualmente, si tenemos alguna empresa o queremos darnos a conocer es recomendable tener un sitio web. Aunque poco a poco las redes sociales han ido ganando terreno a estos, suelen convivir, es decir, una empresa puede tener su sitio web y perfiles en las diferentes redes sociales.

Pese a esto cada vez los sitios webs se vuelven más complejos. No solo a nivel tecnológico. El buen diseño de un sitio web no es una tarea sencilla. La calidad de un sitio web influirá en el usuario final que la visite y posteriormente en la imagen de la empresa.

Con este trabajo se pretende probar diversos sitios web con el fin de encontrar información y problemas en estos. Para ello se analizará el código html, etiquetas, enlaces, contenidos, formularios, y se lanzarán pruebas sobre estos

3 Marco teórico o Estado del arte

Un sitio web está formado por archivos con código de lenguaje de marcas, y otros archivos como imágenes, documentos o scripts. Para acceder a él es necesario conocer su dirección IP pública, o dirección dns(1)

En 1990 Tim Berners-Lee creó la primera versión del lenguaje de marcas Hypertext Markup Languaje HTML además el protocolo Hypertext Transfer Protocol HTTP. Además, se creó el primer navegador web y el primer servidor web.

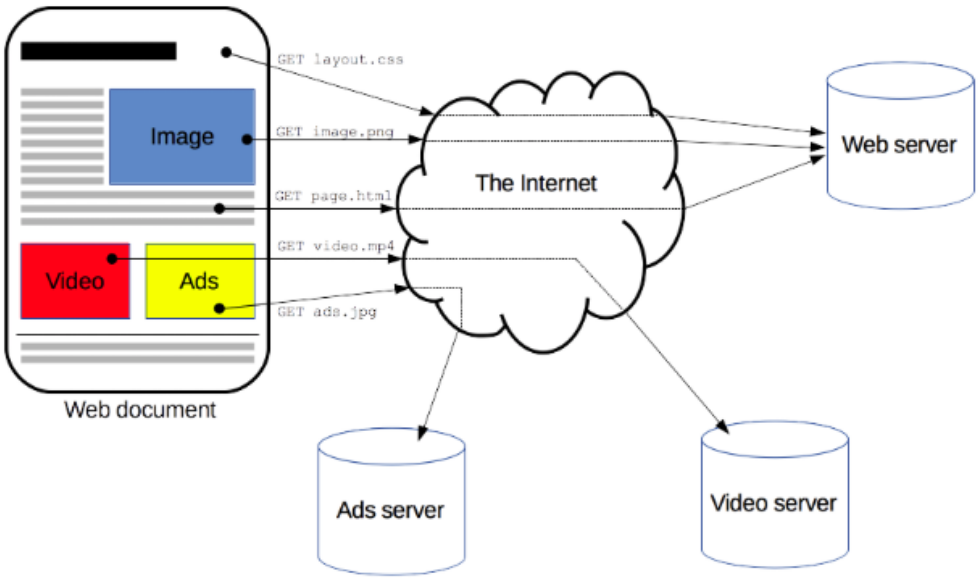


Figura 1: Protocolo HTTP

El protocolo HTTP se apoya en la arquitectura de Cliente-Servidor. Básicamente el funcionamiento consiste en que el cliente envía peticiones al servidor, este la procesa y responde al cliente.

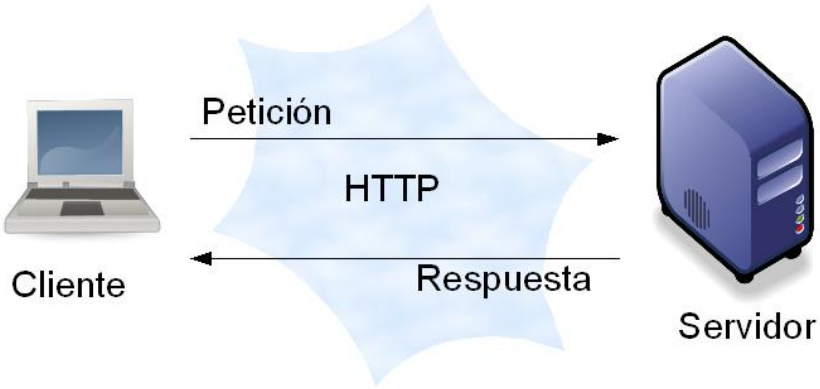


Figura 2: Arquitectura cliente servidor

Esto conlleva a que el cliente envíe la información con un determinado protocolo, para que el servidor sea capaz de leerla e interpretarla, los mismo ocurre con la respuesta del servidor.

Ventajas de esta arquitectura son la simpleza de entendimiento, su escalabilidad, es decir, la facilidad para hacer que el sistema sea capaz de procesar más información

Podrían ser unas desventajas el caso de que muchos clientes envíen una petición al servidor en el mismo instante de tiempo o muy próximo. El servidor tiene que tener la capacidad de atender a un mínimo de clientes sin demasiado retardo y no retrasarse demasiado en responder a estos.

En cuanto al funcionamiento de http tenemos varios tipos de peticiones

- GET: Solicita información o recurso al servidor. Puede incluir parámetros en la url. Esto tiene el inconveniente de que aparecen a simple vista en la barra del navegador.
  `/index.php?page=main&lang=es`

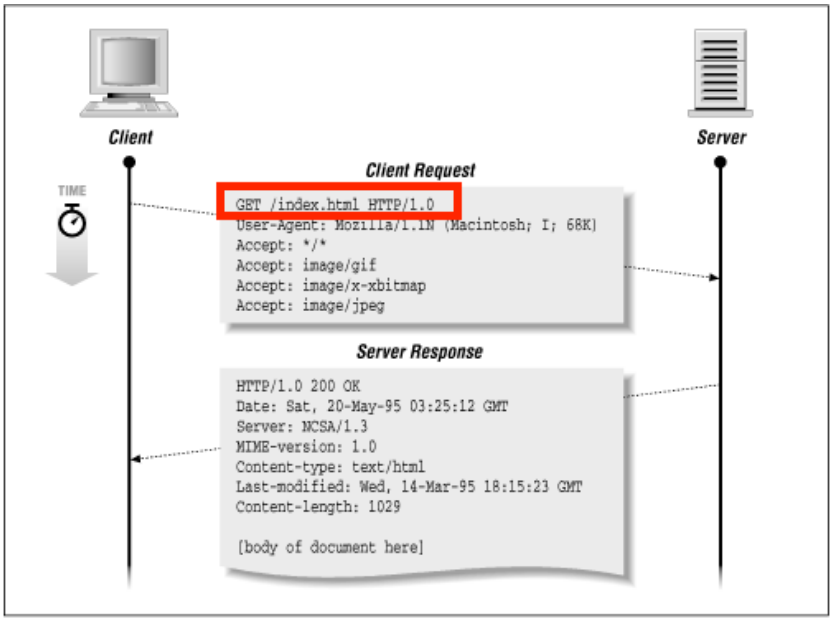


Figura 3: Petición GET

  Como apreciamos en la imagen se pide un recurso /index.html
  El campo `User-Agent` es una forma de decirle al servidor qué navegador somos para que si tiene mecanismos nos devuelva el código HTML que se visualice mejor si se da el caso.
  En la respuesta podemos ver el código de estado de la petición, en este caso 200, la fecha, el tipo de contenido, el tamaño de contenido y el contenido devuelto al final que será un código HTML según el campo `Content-type`

- POST: Envía información al servidor para que este la procese. Es similar a GET pero los parámetros se especifican en el cuerpo de la petición. El servidor también puede devolver información como es el caso de GET

- PUT: Envía un recurso al servidor
- DELETE: Envía la petición al servidor de eliminar el recurso especificado

Los principales códigos de respuesta de una petición HTTP son:

200	La petición se ha llevado a cabo correctamente.
201	Recurso creado.
301	Recurso movido a otra dirección: La nueva se indica en un campo de la cabecera. Normalmente los navegadores web redireccionan a esta dirección nueva.
401	No autorizado: No estamos identificados para el servidor.
403	Prohibido: Normalmente se devuelve cuando se intenta acceder a un recurso que el administrador del sitio no quiere que accedamos
404	Recurso no encontrado: Es el más común, cuando no es encontrado el recurso. A veces cuando estamos navegando y hacemos clic en un enlace se nos retorna este error debido a que el recurso ya no existe o ha cambiado de dirección y al administrador se le ha olvidado modificar el enlace en el código.

Figura 4: Tabla de códigos HTTP

Desde que Tim Berners-Lee creó el primer navegador el auge del del ordenador para uso profesional y el ordenador personal hizo que otras empresas empezaran a desarrollar como Netscape Navigator de Netscape Communicatios que ahora es Mozilla para diversos entornos como Unix, Linux, Windows o los Microsoft Internet Explorer o Apple Safari.

El funcionamiento consiste en hacer una petición GET al sitio web. Para mostrar el HTML, imágenes incluso documentos. La dirección web está formada por el nombre de dominio y la ruta ejemplo http://www.servidor.com/ruta o https://www.servidor.com/ruta

El contenido principalmente de un sitio web HTML, también junto a otros recursos como imágenes o documentos.

Un resumen de las versiones de HTML:

- 2.0: 1995 se publica el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML, es decir, el HTML 1.0 no existió como estándar. HTML 2.0 no soportaba tablas. Se simplificaba al máximo la estructura del documento para agilizar su edición, donde la declaración explícita de los elementos body, html y head es opcional.
- 3.2: 1997 y es la primera recomendación de HTML publicada por el W3C (World Wide Consortium). Esta revisión incorporó los últimos avances de las páginas web desarrolladas hasta 1996, como applets de Java y texto que fluye alrededor de las imágenes.
- 4.01: 1999 La última especificación oficial del W3C. Después, el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (Web Hypertext Application Technology Working Group) que comenzó el desarrollo del HTML 5, cuyo primer borrador oficial se publicó en enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de

2007 el W3C decidió retomar la actividad estandarizadora de HTML, dentro del cual decidió integrar el XHTML.

- 4.01 Strict: En este tipo no se aceptan etiquetas obsoletas. Es la versión que si usamos en teoría nos debería dar un resultado óptimo en los navegadores más modernos.
- 4.01 Transitional: En este tipo de documentos se pueden usar todas las etiquetas de todas las versiones de HTML. Usar esta variante de HTML plantea el interrogante de si es correcto permitir el uso de etiquetas obsoletas que podrían dejar de funcionar en las próximas versiones de los navegadores. Podemos tener a una mejor visualización en la mayor parte de los navegadores.
- 4.01 Frameset: Tiene soporte para frames. Los frames son unos marcos a modo de pequeñas subventanas dentro de una misma página web que se usaban mucho hace unos años pero que hoy en día se usan cada vez menos.
- 5: 2014: El consorcio internacional W3C, después de una evolución de varios años. HTML 5 incorpora nuevos elementos. Se introduce la posibilidad de introducir audio y video de forma directa en la web sin necesidad de plugins o complementos en los navegadores, y otras novedades. El W3C irá lanzando progresivamente nuevas evoluciones del HTML 5.

3.1 Herramientas existentes

En esta sección se describirán las herramientas para analizar sitios web ya existentes. Estas herramientas son conocidas por parte de los desarrolladores web.

Firebug: Es una extensión de Mozilla Firefox que permite: Ver el HTML de la página web, ver el output de la consola de Javascript del navegador, editor de estilos, visualizador de peticiones http

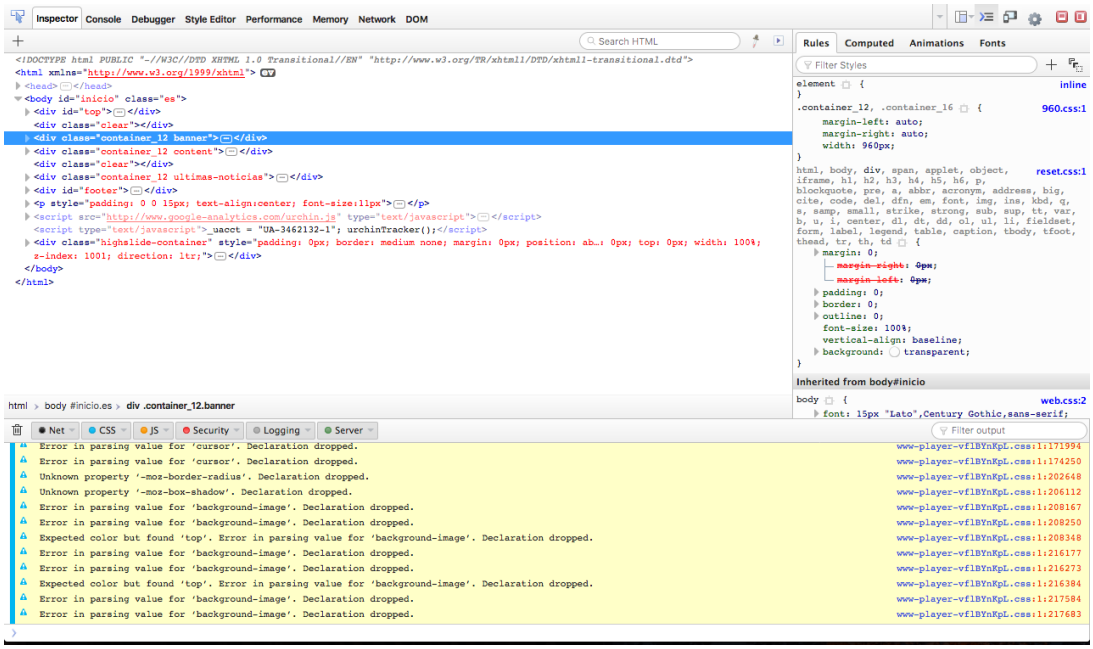


Figura 5: Visualizador del código HTML de Firebug

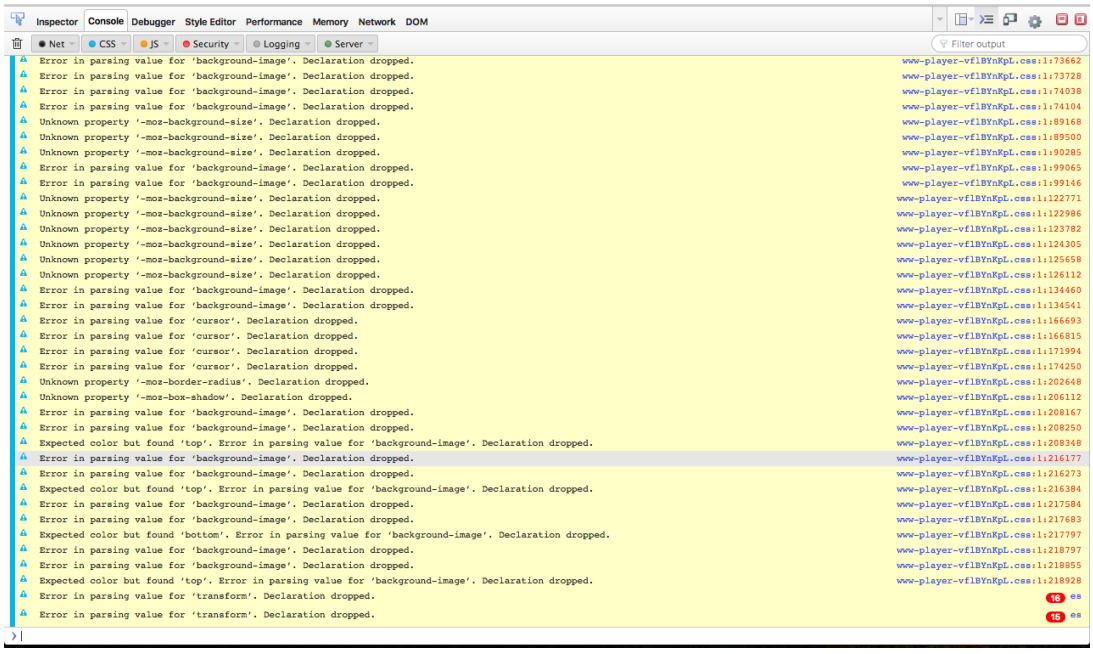


Figura 6: Visualizador de la consola de Javascript del navegador de Firebug

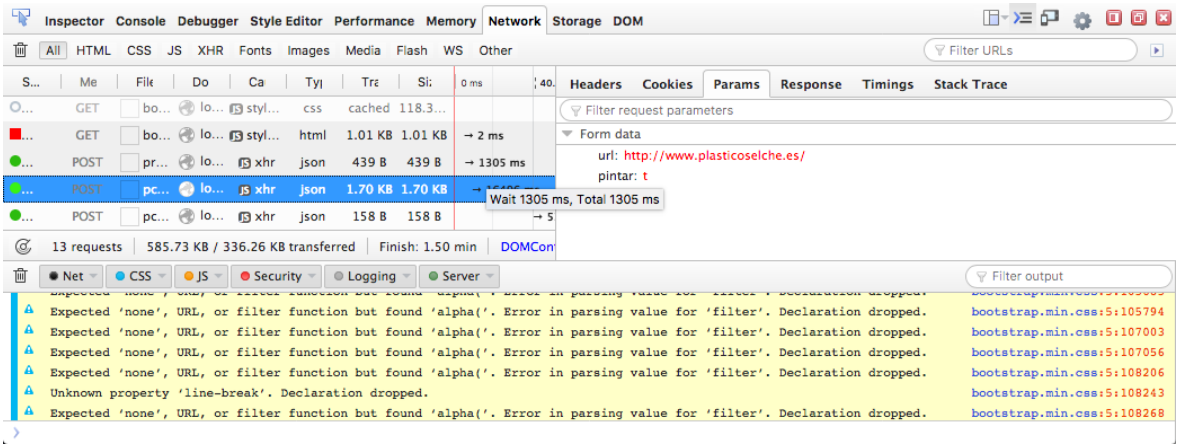


Figura 7: Visualizador de las peticiones HTTP de Firebug

Curl: Herramienta de línea de comandos generalmente orientada a la transferencia de archivos. Permite hacer peticiones HTTP, dada la dirección del sitio. Es posible usarla como biblioteca en algunos lenguajes de programación como se verá a continuación. Dos ejemplos sencillos de petición GET y petición POST se ven a continuación.

```
curl http://www.example.org:1234/
```

```
curl -X POST --data "campo1=valor1&campo2=valor2"
"http://www.ejemplo.com/"
```

```
curl -X POST --data
"kt_login_user=value2&kt_login_password=value2"
http://www.reformasenelche.es/contacta-con-nosotros/#wpcf7-f53-p14-o1
```

Apache Bench: Herramienta de línea de comandos disponible en los Unix y Gnu/Linux. Software que está diseñado para medir el rendimiento de los sitios web. Permite hacer peticiones de forma concurrente hacer pruebas de carga a cualquier sitio web

```
ab -n 1000 -c 100 http://localhost:4567/
```

Esto significa que se harán 1000 peticiones y después 100 peticiones de forma concurrente.

Podemos hacer peticiones POST y ver resultados de la ejecución con el modo –v

```
ab -n 100 -c 5  -T 'application/x-www-form-urlencoded' -p post.txt
-v 2 http://google.com/
```

Apache Jmeter: Herramienta en Java. Al igual que Apache Bench se utiliza para medir el rendimiento de un sitio web con pruebas de carga. Además tiene aserciones y se puede almacenar datos en el disco duro, y también verlos en un gráfico mediante.

PHPUnit: Al estilo de jUnit en el lenguaje Java tenemos PHPUnit. Este entorno se utiliza para hacer pruebas unitarias a clases de PHP. Para la gran cantidad de lenguajes de programación existe un entorno o framework para hacer pruebas unitarias. SUnit de Kent Beck para Smalltalk fue la primera suite para un lenguaje de pruebas unitarias del estilo.

PhantomJS y **CasperJS**: Herramientas en que utilizan NodeJS.+Básicamente es un navegador sin interfaz gráfica. Está hecho sobre NodeJS. Esta herramienta simula el navegador web. Además se puede automatizar navegación, tomar capturas de pantalla y hacer aserciones. Se utiliza para hacer pruebas funcionales a un sitio web. Empresas que usan PhantomJS:

- Twitter usa QUnit y PhantomJS
- LinkedIn
- Netflix usa Sketchy, basado en PhantomJS
- Time Warner Cable usa PhantomJS con CoffeeScript , Jasmine, y Junit XML para la integración continua con Jenkins

```
casper.start(url);

casper.then(function() {
  primero = this.getPageContent();
  imagenes = imagenes.concat(formularioNombre+'0.jpg\n');
  this.captureSelector(formularioNombre+'0.jpg', 'form');

  this.fill(dir,inputs,false);
  this.click('input[type="submit"][name="'+ submitName +
});

casper.on('remote.alert', function(message) {
      mensajeAlert = 'Mensaje de alerta: ' + message;
});

casper.then(function(){
  segundo = this.getPageContent()
});

casper.run(function(){
  this.captureSelector(formularioNombre+'1.jpg', 'form');
  imagenes = imagenes.concat(formularioNombre+'1.jpg')
```

Figura 8: Script CasperJS

Selenium IDE: Se utiliza para hacer pruebas funcionales también. Es un complemento de Mozilla Firefox donde se introducen las órdenes de la navegación web como scripts, ya que cuenta con su lenguaje específico de dominio. En la imagen podemos ver: cómo se abre la la página web, se introduce en el cuadro de texto q el un valor, se hacen varios clicks, y finalmente se comprueba una aserción.

Selenium Web Driver: Similar al anterior pero se integra el código fuente. Se utiliza un controlador específico para cada navegador. Cuando se inicia una suite de pruebas este instancia y se ejecutan en

el navegador. Java, Ruby, Python y C# son los lenguajes para los que está implementado. También hay una implementación para PHP de Facebook disponible en Github.

Análisis SEO: Que significa Search engine Optimization. Los buscadores como Google, Bing, Yahoo tienen mecanismos para elegir el orden de los resultados que aparecen en sus búsquedas. De todos los usuarios que acceden a nuestro sitio web la gran mayoría lo hacen desde un buscador. Esto es así porque el usuario recuerda mejor unas cuantas palabras o palabra clave para encontrar un sitio web que recordar la url entera completa es más complicado. A continuación se explicarán algunas de las medidas para posicionar mejor un sitio web entre los buscadores y de cara al usuario final.

Tener solo una etiqueta: `<h1>, <title>` y no muy largas con el total de caracteres menor a setenta. La etiqueta `<title>` es la que nos aparece en los resultados de búsqueda del buscador como enlace, también nos aparece en la barra del navegador cuando estamos visitándola. La etiqueta `<h1>` va asociada al título de la página pero en el contenido de esta. En cuanto a la `<h2>` referencia las diferentes partes de la página, tiene que estar relacionadas con el contenido que describe.

Etiquetas `<meta>`. Estas se encuentran en la parte de cabecera, es decir, dentro de la etiquetas `<head>`. Antes los buscadores se guiaban más por estas etiquetas que ahora debido a que es muy fácil para el desarrollador insertar demasiada información y así aparecer en más resultados de búsqueda aunque no estén relacionados con esta.

- `Description`: Pequeño resumen del sitio web. Se utiliza por los buscadores como resumen en sus páginas de resultado. Por ejemplo cuando un usuario está buscando en Google puede ver la descripción del sitio web.
- `Keywords`: Palabras que tienen relación con el sitio web,
- `Robots`: Se explicará más abajo.
- `Viewport`: Se usa en los sitios que cuentan con una versión para móviles de adaptable o responsive. Sirve para indicar cómo de grande se mostrará la página en la pantalla.
- `Name=”google” content=”Nosnippet”:` Para evitar que el contenido de tu sitio web aparezca en los fragmentos destacados de una búsqueda con google
- `Name=”google” content=”Notranslate”:`
- `http-equiv="Content-Type" content="text/html; charset=utf-8"`
- `Author, Subject, Generator, Language, Revisit-after`

Valores `alt=””` en las etiquetas de imágenes `<img>`. Comprueba, por ejemplo, si hay imágenes en tu publicación, si tienen este valor y este contiene la palabra clave objetivo de esa publicación.

Google comprueba si tus publicaciones son lo suficientemente largas, si has escrito una `meta description` y si esa meta `description` contiene tu palabra clave objetivo, si necesitas algún subtítulo dentro de tu publicación, etc,

Google ha anunciado recientemente que deberíamos utilizar los elementos de enlace `rel="next"` y `rel="prev"` en la sección `<head>` de tus archivos con paginación.

Por la parte de mostrar el mejor contenido de nuestro sitio web tenemos el archivo `robots.txt` y etiquetas meta robots.

Los buscadores web cuando quieren rastrear sitios web para mostrarlos en los resultados de sus búsquedas se fijan primero en el archivo `www.servidor.dominio/robots.txt`. Este archivo indica al buscador qué recursos del sitio web quiere que se vean y cuáles no. Funciona con una serie de reglas y comodines.

Cualquier robot rasteará el sitio	• `User-agent:*` • `User-agent: Googlebot` • `User-agent: Bingbot`
Permitir acceso a	• `Allow: /`
Dengar acceso a	• `Disallow: /privado/` • `Disallow /*.png$`
Indicar ruta donde está el mapa del sitio xml	• `Sitemap: http://www.servidor.dominio./sitemap.xml`
Número de segundos que debe esperar entre página y página. Para que no se sobrecargue nuestro servidor	• `Crawl-delay: 30`
(*): Cualquier secuencia de caracteres	• `/privado*/`
($): El final de una url	• `/*.php$`

Figura 9: Reglas archivo `robots.txt`

Hay que resaltar, que aunque se deniegue algún recurso de ser indexado, no evita acceder a él si sabemos su nombre y accedemos con el navegador.

Los Disallow en el archivo robots.txt no solo sirven para que un navegador no indexe partes no importantes de tu sitio web, como imágenes, archivos `.css`, `.js`, `.php`. También es recomendable debido a que los robots de los buscadores tienen un número máximo de direcciones que rastrear por sitio web. Les facilitaremos la tarea y esto hará que seamos bien puntuados por el buscador.

Meta robots. Similar al archivo anterior pero deben de estar en las etiquetas meta de cada página de nuestro sitio web. La diferencia es que el sitio no será indexado, pero sí rastreado, siguiendo los enlaces hacia otras páginas y transmitiendo el valor de esos enlaces. El contenido deberá estar en las etiquetas dentro de `<head> </head>`

Ejemplo: `<meta name=”robots” content= “NoIndex,Follow”>`

- `Index/NoIndex`: Si tu página debe incluirse en los índices del buscador
- `Follow/NoFollow`: Si los enlaces de tu página deben rastreados y pasar el valor de la página o no

Las combinaciones más comunes son:

- `Index,Follow`: Se permite la indexación y rastreo
- `NoIndex,Follow`: No se indexa pero sí que se rastrea. Es buena combinación si deseas que tu página no aparezca en el índice de los buscadores.
- `Index, NoFollow`: Permite indexación pero evita el rastreo. Es buena cuando quieres que aparezca en los buscadores y evitar que se rastree. Por ejemplo, contenido generado por el usuario de la web.
- `NoIndex, NoFollow`

4 Objetivos

Analizando el problema podemos ver que se puede desarrollar un software de escritorio, o una aplicación web. Se toma la decisión de hacer la versión web, de esta forma habrá menos incompatibilidades para utilizarlo en otras plataformas teniendo menos requerimientos para su utilización. Pues todos tienen un navegador web. Se requerirá el lenguaje en el que se desarrolle junto con alguna librería y poco más. Se decide elegir el lenguaje PHP. Aunque no se tiene mucha experiencia previa es un lenguaje maduro y que se han resuelto gran cantidad de problemas con él. Además se elige por su sencillez y flexibilidad.

A partir de una url se hará una petición y a partir de su código HTML se analizará el sitio obteniendo:

Enlaces internos: Desde el sitio web, a qué direcciones web dentro del mismo servidor podemos acceder.

Enlaces externos: Desde el sitio web, a qué direcciones web fuera del servidor podemos acceder.

Enlaces rotos: Desde el sitio web, a qué direcciones web intentamos acceder pero no es posible. Debido a que ha cambiado la dirección o ya no existe (404).

Enlaces a contenidos: Como pueden ser imágenes, hojas de estilo `css`, documentos `pdf`, `scripts`

Formularios: Los formularios que existen en el sitio, para ello nos fijaremos en la etiqueta `<form`. Para ello se buscarán en todos los enlaces del sitio web

Pruebas a formularios: El objetivo es probar el envío de formularios con varios valores, incluso con erróneos

- Servidor: Se enviarán peticiones POST con diferentes valores al servidor directamente.
- Cliente: Se intentará hacerse pasar por el navegador web, para ver los mecanismos en la parte del cliente `scripts` que impiden el envío de campos incorrectos al servidor.
- Carga: Se enviarán varias peticiones concurrentemente para ver parámetros como el número de peticiones por segundo que soporta el sitio, tiempo que emplea el sitio en atender una petición.

Análisis HTML: Analizando el contenido del HTML, se analizarán etiquetas que ayudan a los motores de búsqueda a encontrar el sitio web introduciendo palabras relacionadas

De esta forma se obtendrá una visión general sobre el estado del sitio web y sus fallos más importantes a simple vista

5 Metodología

Para llevar a cabo el trabajo se seguirá una metodología de reuniones periódicas con el tutor. Estas reuniones serán entre 2 semanas mínimo. En cada reunión se revisa el trabajo hecho hasta ahora y se define la siguiente fase, además se resuelven dudas respecto a la etapa anterior o se corrigen puntos anteriores. También se resuelven dudas puntuales con el tutor mediante email. A continuación se muestra el documento que se ha ido construyendo junto al tutor

- 15-02-2017
    - Definir las características del sistema: Aplicación web, Pruebas de cliente, Estudiar herramientas actuales: `PhantomJS`, `PHPUnit`
    - Definir las prestaciones del sistema: Mapa del web, Enlaces rotos, Formularios, Seguridad
- 15-03-2017
    - Implementar un 'holamundo':
    - Estudiar las formas que hay de obtener contenido remoto vía HTTP con PHP, diferencias entre ellas y ventajas/inconvenientes.
    - Implementar un script PHP que acceda a una URL y obtenga ciertos datos del resultado obtenido: tiempo respuesta, versión HTML, contenido, resultado, etc.
- 25-04-2017
    - Agrupar código en funciones, versión HTML, parte con `file_get_contents()`
    - Niveles navegación, mapa de niveles(textual de momento), enlaces rotos, enlaces externos, enlaces internos.
    - Formularios que hay, analizar, cuenta de campos que hay
- 02-05-2017
    - Pruebas de servidor (enviando POST con `Curl`, etc.)
    - Enviar POST y ver las respuestas e informar del comportamiento
    - Pruebas de cliente (emulando usuarios que rellenan formularios (`PhantomJS`).
    - Intentar envíos con datos incorrectos en los campos del formulario, ver cómo hacer pruebas sobre el formulario original.
    - Decorar los resultados con alguna plantilla con gráficas (`Highcharts`)
- 02-06-2017
    - Pruebas 'extremas': otros recursos (`API REST`, `Rss`), acceso 'no legal' a recursos (imágenes, documentos), estrés, capacidad, urls no existentes, métodos incorrectos
    - Calidad del HTML (a partir de estadísticas de tags)
    - Temas SEO, a partir de contenidos y otros indicadores.

6 Cuerpo del trabajo

6.1 Desarrollo

Fase 1:

Se hace la primera reunión entre el tutor y el alumno para acordar características del Sistema. Primero se acuerda si va a ser una aplicación web o de escritorio y su respective plataforma y lenguaje. Son multitud e los lenguajes que se han visto a lo largo del grado. Como `Java, C++, C#, Scala, Php, Javaspript` en `NodeJS.`

Decidimos hacer un sitio web por su flexibilidad, ya que se necesita poco más que el servidor web, navegador y el lenguaje en cuestión para ponerlo en funcionamiento.

También se decide hacia dónde enfocar el trabajo. Si analizar el sitio web desde la parte del servidor o la parte de cliente. Se decide que desde la parte de cliente. Como usuarios de un sitio web tenemos acceso solo a las respuestas del servidor.

El trabajo puede ir enfocado a encontrar los enlaces del sitio `<a href=""` y comprobar su estado. Hacer un mapa de la web con estructura de árbol a partir de los enlaces, analizar los formularios que tiene el sitio y posteriormente probarlos.

También se planifica la segunda fase del proyecto acordando llegar a una serie de hitos.

Fase 2

El alumno a trabajar por su cuenta Se habían planteado los puntos de:

- Implementar un 'holamundo':
- Estudiar las formas que hay de obtener contenido remoto vía HTTP con PHP, diferencias entre ellas y ventajas/inconvenientes.
- Implementar un script PHP que acceda a una URL y obtenga ciertos datos del resultado obtenido: tiempo respuesta, versión HTML, contenido, resultado, etc

Se empieza por estudiar las diferentes formas de obtener el HTML de un sitio web con el lenguaje PHP. Para obtener información de una URL tenemos dos formas

Función `file_get_contents()`: Esta función lee el documento con el nombre pasado por parámetro. Puede usarse tanto para ficheros en el disco duro local o direcciones web.

Esta función no funciona para extraer el contenido de algunos sitios web, algunos proveedores de hosting no permiten su funcionamiento. Otra desventaja es que solo nos devuelve el contenido, sea `html`, `txt`, `png`.

Curl; Desde la versión de PHP 4.0.2. Se ha portado la biblioteca de Daniel Stenberg libcurl. Esta librería hace lo mismo la versión de intérprete de comandos. Tenemos que indicar parámetros como la url, agente de usuario, cabeceras que queremos enviar, ssl por si nos conectamos a HTTPS

```
$this->ch = curl_init();
curl_setopt($this->ch,CURLOPT_URL,$this->url);
curl_setopt($this->ch,CURLOPT_USERAGENT,$this->userAgent);
curl_setopt($this->ch,CURLOPT_HTTPHEADER,array("Accept-Language: es-es,en"));
curl_setopt($this->ch,CURLOPT_TIMEOUT, 10);
curl_setopt($this->ch,CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($this->ch,CURLOPT_RETURNTRANSFER, 1);
curl_setopt($this->ch, CURLOPT_SSL_VERIFYPEER, 0);
$this->initTime = microtime(true);
$this->result = curl_exec($this->ch);
$this->endTime = microtime(true);
```

Figura 10: Curl en PHP

Para saber la versión de HTML se utiliza un array asociativo y se comprueba qué valor tiene la primera línea del HTML

```
$this->versionesHTML =[
  "<!DOCTYPE html>" => "html5",
  "-//W3C//DTD HTML 4.01//EN" => "html401Strict",
  "-//W3C//DTD HTML 4.01 Frameset//EN" => "html401Frameset",
  "-//W3C//DTD XHTML 1.0 Strict//EN" => "xhtml101Strict",
  "-//W3C//DTD XHTML 1.0 Transitional//EN" => "xhtml101Transitional",
  "-//W3C//DTD XHTML 1.0 Frameset//EN" => "xhtml10Frameset",
  "-//W3C//DTD XHTML 1.1//EN" => 'xhtml11',
];
```

Figura 11: Versiones HTML

Expresiones regulares: El lenguaje PHP también cuenta con la posibilidad de filtrar contenido de una variable mediante expresiones regulares. En PHP se pueden usar las expresiones compatibles con PERL. Tenemos el siguiente ejemplo que devuelve el valor de la etiqueta pasada por parámetro.

```
function getTitle(){
  return $this->getElementValue("title");
}

function getElementValue($element){
  preg_match_all("(<$element>(.*)<\/$element>)siU",$this->result,$coincidencias);
  return $coincidencias[1][0];
}

function getContentBetweenTags($tagName,$label,$content){
  if($label == 1){
    $pattern = "/".$tagName."/im";
  }
  else if ($label == 2){
    $pattern = "|<\s*?".$tagName."\b[^>]*>(.*?)</".$tagName."\b[^>]*>|s";
  }

  else if ($label == 3){
    $pattern = "|<".$tagName."(.*)>|U";
  }

  preg_match_all($pattern, $content, $coincidencias);
  //preg_match_all($pattern, $content, $coincidencias);
  return $coincidencias[0];
}
```

Figura 12: Función expresiones regulares

De esta forma se extrae el título de la página html. Se comprueba el contenido entre las etiquetas `<title></title>`. Podemos ver en la función la idea de filtrar otro tipo de etiquetas

Como vemos, después de la ejecución, la clase nos devuelve varios datos. Hacemos la petición y hacemos que nos devuelva los datos más importantes en un array:

```
Array
(
    [0] => Array
        (
            [key] => Url efectivo
            [value] => https://www.google.es/?gfe_rd=cr&dcr=0&ei=I4CqWfP_GaLBwgLDrZ2YDQ&gws_rd=ssl
        )

    [1] => Array
        (
            [key] => Titulo
            [value] => Google
        )

    [2] => Array
        (
            [key] => CÃ³digo de respuesta
            [value] => 200
        )

    [3] => Array
        (
            [key] => Tipo de contenido
            [value] => text/html; charset=UTF-8
        )

    [4] => Array
        (
            [key] => Total time
            [value] => 1.5124049186707
        )

    [5] => Array
        (
            [key] => VersiÃ³n del HTML
            [value] => html5
        )

)
```

Figura 13: Datos básicos de la petición

Fase 3

Los objetivos de esta fase son

- Agrupar código en funciones, versión html.
- Niveles navegación, mapa de niveles(textual de momento), enlaces rotos, enlaces externos, enlaces internos.
- Formularios que hay, analizar, cuenta de campos que hay.

Librería `Simple HTML DOM` para PHP. Esta librería nos simplifica la tarea de extraer información del fichero HTML. La idea inicial era extraer información del código HTML utilizando expresiones regulares. Al descubrir esta librería que simplifica enormemente el trabajo cambiaron las funciones. La función que extrae los links del HTML es ésta, se comprueba si es una página ya que podría ser un documento con alguna extensión como: `pdf, png` ...

```php
$html = str_get_html($this->getResult());
foreach($html->find('a') as $link) {
  $linkN = $link->href;
  if($this->isWebPage($linkN)){
    if(!in_array($linkN,$this->links)){
      array_push($this->links,$linkN);
    }
  }
}
$this->links = array($this->url => $this->links);
```

Figura 14: Función que extrae los links

Las funciones para comprobar si es un link interno o externo comprueban si el link contiene la cadena del servidor. En caso de que sea un link completo se mira si contiene la subcadena de su servidor. Dependiendo del web se encuentran las rutas de la web sin o con el servidor. Para comprobar externos se mira que no contenga la subcadena del servidor y que contenga la subcadena `http` o `www`. Finalmente en los links rotos se hace una petición GET y se comprueba que el código de respuesta sea múltiplo de 400.

Para extraer la estructura de árbol se utiliza un array con el valor de la página actual, el segundo componente es una array que contiene primero la página desde donde ha sido accedido ese link, y los siguientes links.

```
        [interno] => www.plasticoselche.es/es/noticias/20/mood-2013/
    )

[18] => Array
    (
        [interno] => www.plasticoselche.es/nota.php
    )

[19] => Array
    (
        [externo] => http://www.mediaelx.net
    )

[20] => Array
    (
        [http://www.plasticoselche.es/] => Array
            (
                [0] => primero
                [1] => Array
                    (
                        [interno] => www.plasticoselche.es/es/
                    )

                [2] => Array
                    (
                        [interno] => http://www.plasticoselche.es/es/
                    )

                [3] => Array
                    (
                        [interno] => http://www.plasticoselche.es/en/
                    )

                [4] => Array
                    (
                        [interno] => www.plasticoselche.es/es/sobre-nosotros/
                    )

                [5] => Array
                    (
                        [interno] => www.plasticoselche.es/es/catalogo/
                    )

                [6] => Array
```

Figura 15: Array con los valores de los links internos, externo y el árbol

En la parte de extraer formularios la primera versión de la función extraía el html y lo pintaba directamente en la pantalla. Posteriormente se tuvo que modificar.

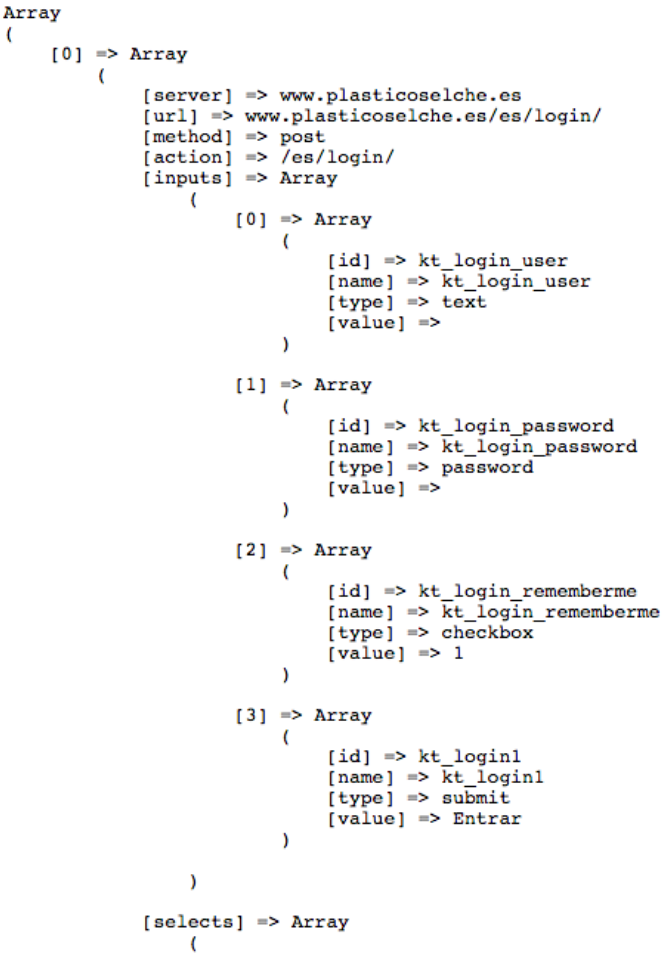


Figura 16: Array que muestra la información de un formulario

Primero se busca en los links internos del sitio web. Una función va comprobando en cada link, esta llama otra que extrae los formularios mirando el contenido entre las etiquetas `<form></form>`.

Se comprueba que el formulario no sea igual que alguno anterior aunque se encuentre en otro link.

Se analiza el contenido extrayendo los campos principales como el método, `url`, `action` y los diferentes campos de entrada de información como:

- `inputs`: `text`, `password`
- `selects`
- `textareas`

Creando un array que contiene: `method`, `action`, `url`, inputs con `id`, `name`, `type`, `textareas` `selects` con su `id` y los diferentes valores para después representarlo en el HTML y hacer pruebas.

Fase 4

- Pruebas de servidor (enviando POST con CUrl, etc.)
- Enviar POST y ver las respuestas, informar del comportamiento
- Pruebas de cliente (emulando usuarios que rellenan formularios (PhantomJS).

- Intentar envíos con datos incorrectos en los campos del formulario, ver cómo hacer pruebas sobre el formulario original.
- Decorar los resultados con alguna plantilla con gráficas (Highcharts)

Pruebas de servidor: Se utiliza el mismo CURL de las peticiones GET. Ahora se hace la petición con diferentes parámetros parámetros.

```
function curlPost($url,$datosBody,$method){
  $postdata = http_build_query($datosBody);
  $ch = curl_init();
  curl_setopt($ch,CURLOPT_URL,$url);
  curl_setopt($ch,CURLOPT_USERAGENT,'$this->userAgent');
  curl_setopt($ch,CURLOPT_HTTPHEADER,array(
  'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0',
  'Accept-Language: es-es,en;q=0.5',
  'Content-Type: application/x-www-form-urlencoded',
  "Accept: text/plain"
  ));
  curl_setopt($ch,CURLOPT_CUSTOMREQUEST,strtoupper($method));
  curl_setopt($ch,CURLOPT_TIMEOUT, 10);
  curl_setopt($ch,CURLOPT_POST,true);
  curl_setopt($ch,CURLOPT_FOLLOWLOCATION, 1);
  curl_setopt($ch,CURLOPT_POSTFIELDS,$postdata);
  curl_setopt($ch,CURLOPT_HEADER,true);
  curl_setopt($ch,CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
  curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);

$initTime = microtime(true);
  $result = curl_exec($ch);
  $endTime = microtime(true);
```

Figura 17: CURL para POST

Parámetros como los datos, donde se le pasa un array asociativo con el nombre del input y el valor. Resaltar también el valor de las cabeceras. Le indicamos que aceptamos HTML, le enviamos contenido de formulario etc. El resultado de la petición la vemos aquí abajo.

```
PeticiÃ³n post a http://www.plasticoselche.es/es/login con datos:

Array
(
    [kt_user_login] => abc
    [kt_login_password] => ava
)
1

En 2.6539821624756 con cÃ³digo de respuesta: 200

Login error
```

Figura 18: Resultado petición POST

En todos los sitios que se ha probado es normal que se devuelva el código 200 de error ya que no se requiere estar autentificado. Aunque se envíe un campo de menos el servidor lo acepta y devuelve un 200. Más adelante veremos el mecanismo del sitio web para recibir datos no erróneos.

Pruebas de cliente: Para las pruebas de cliente se buscó información sobre `PhantomJS` aunque en `NodeJS`, se puede utilizar en el proyecto indirectamente. Hay una librería de PHP que usa `PhantomJS` y hace lo mismo que este. Sin embargo requiere preparar el entorno para utilizarla. Al final se usa `CasperJS`. Que hace lo mismo que `PhantomJS`. El inconveniente es que desde PHP

hay que agregar el `PATH` donde se encuentran los binarios de estos en nuestro equipo local y la orden que queremos.

```
322
323  function pruebaCliente($inputs){
324    $rutaFichero = $_SERVER["DOCUMENT_ROOT"]."/pruebas/PruebasCliente/pruebasCliente.json";
325    $file = fopen($rutaFichero, "w") or exit("Unable to open file!");
326    fwrite($file,json_encode($inputs));
327    fclose($file);
328    putenv("PHANTOMJS_EXECUTABLE=/usr/local/Cellar/phantomjs/2.1.1/bin/phantomjs");
329    system('/usr/local/Cellar/casperjs/1.1.4-1/bin/casperjs PruebasCliente/casper.js',$var);
330  }
```

Figura 19: Ejecución de CasperJS desde PHP

Los datos se reciben mediante un archivo `JSON` en disco que procesa la consola `CasperJS` cuando la invocamos. `CasperJS` como ya hemos visto es una especie de mini navegador web. Esta procesa los datos, primero abre la dirección url del formulario y hace una captura de pantalla de la sección `<form>`. Se rellena el contenido del formulario con los valores del archivo leído y se hace click en el botón que envíe el formulario. Este script guarda las capturas en un directorio formado por el nombre de la url más la ruta del formulario guardándolo en un archivo de texto.

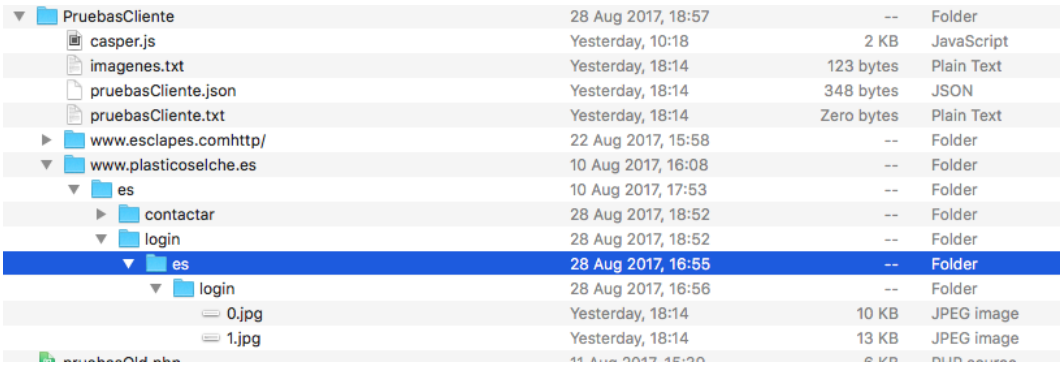


Figura 20: Archivos de texto e imágenes generadas por CasperJS

Fase 5

- Acceso a recursos (imágenes, documentos), estrés, capacidad.
- Calidad del HTML (a partir de estadísticas de tags)
- Temas SEO, a partir de contenidos y otros indicadores.

  En cuanto a los contenidos se obtienen los links y se comprueba si tiene extensión, también se buscan los elementos con etiqueta `<link>` y `<img>` ya que contienen imágenes, hojas de estilo, etc…

  Se filtra el contenido que quede entre etiquetas

  - `<link>` Típicamente hojas de estilo CSS
  - `<a>` Direcciones web pero que no sean páginas, sino documentos como pdf, png
  - `<img>` Imágenes

Además se envía una petición GET al recurso comprobando el código de respuesta

En cuanto a las pruebas de carga, se envían peticiones concurrentes al mismo sitio web con el programa de `Shell Apache Bench`. El procedimiento es el mismo que para ejecutar la consola de `CasperJS`. Podemos especificar el número de peticiones que queremos en total y el número de peticiones que queremos que lance la aplicación automáticamente.

Para el análisis SEO se extraen las etiquetas del sitio principal del HTML se saca un recuento de la cantidad de veces que aparecen en la página, Etiquetas de `título`, `h1`, `h2`, valores `alt` de etiquetas de imagen y por último el archivo `robots.txt` si el sitio web lo tiene. También se extraen la etiquetas meta de las cabeceras y sus respectivos valores

```
Array
(
    [0] => Array
        (
            [key] => html
            [value] => 1
        )

    [1] => Array
        (
            [key] => head
            [value] => 1
        )

    [2] => Array
        (
            [key] => meta
            [value] => 6
        )

    [3] => Array
        (
            [key] => title
            [value] => 1
        )
```

Figura 21: Cantidad de veces que aparece una etiqueta en el HTML

```
Array
(
    [0] => Array
        (
            [name] =>
            [content] => text/html; charset=utf-8
        )

    [1] => Array
        (
            [name] => description
            [content] => PlÃ¡sticos Elche S.A.
        )

    [2] => Array
        (
            [name] => keywords
            [content] => Sinteticos, polipiel, eskai, piel sintetica, imitacion piel, PVC,
termovirantes
        )

    [3] => Array
        (
            [name] => author
            [content] => mediaelx
        )

    [4] => Array
        (
            [name] => viewport
            [content] => width=device-width
        )

)
```

Figura 22: Etiquetas meta con sus valores

6.2 Resultado en el sitio web y prueba de la aplicación

Para representar los resultados de los scripts PHP se ha utilizado código HTML,. Para representar las diferentes secciones del sitio web se han utilizado botones enlazados a funciones `JQuery`. Pues solo con HTML no era suficiente para tratar los datos enviados y recibidos del usuario y representarlos. Se han tenido que utilizar funciones que recogen los datos, ya sean de formulario o de control para enviarlos a PHP y hacer así más fácil la elección de la tarea que hacer. Además se ha usado la librería `Bootstrap` como hoja de estilos css.

Resultado en la web http://www.plasticoselche.es

Links:

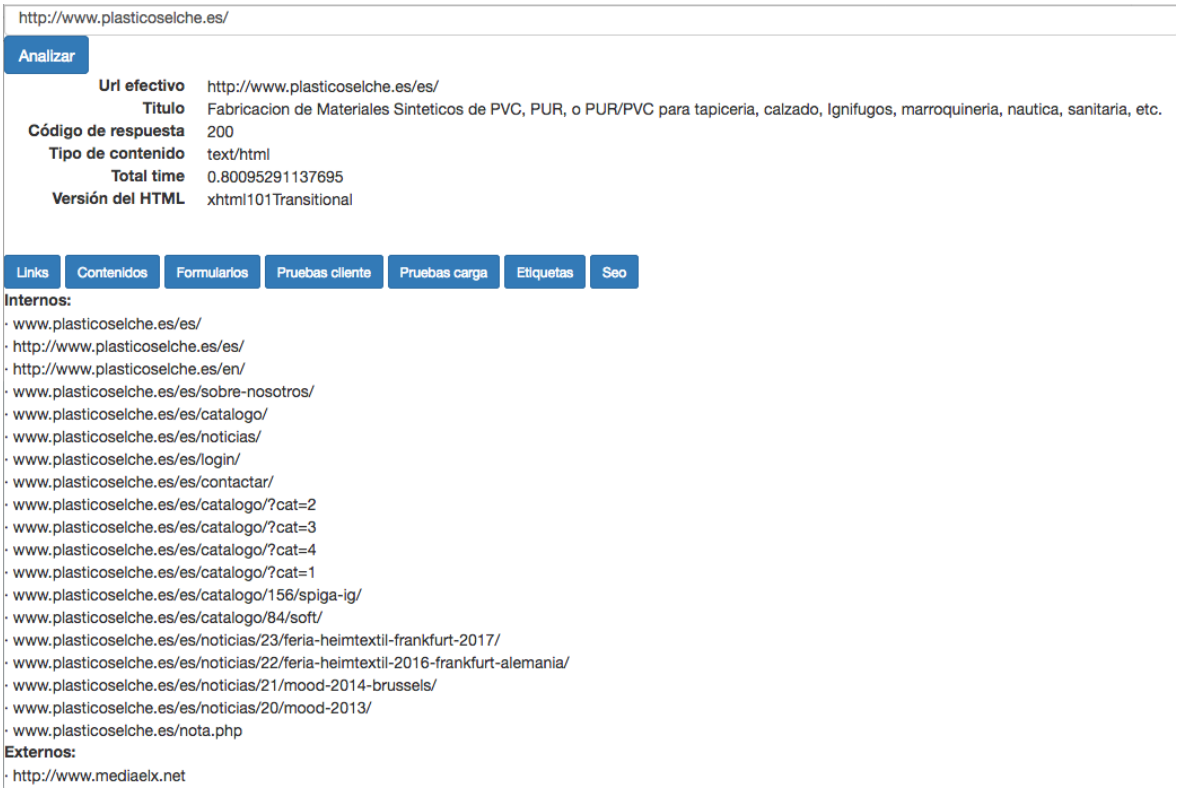


Figura 23: Links

Contenidos:

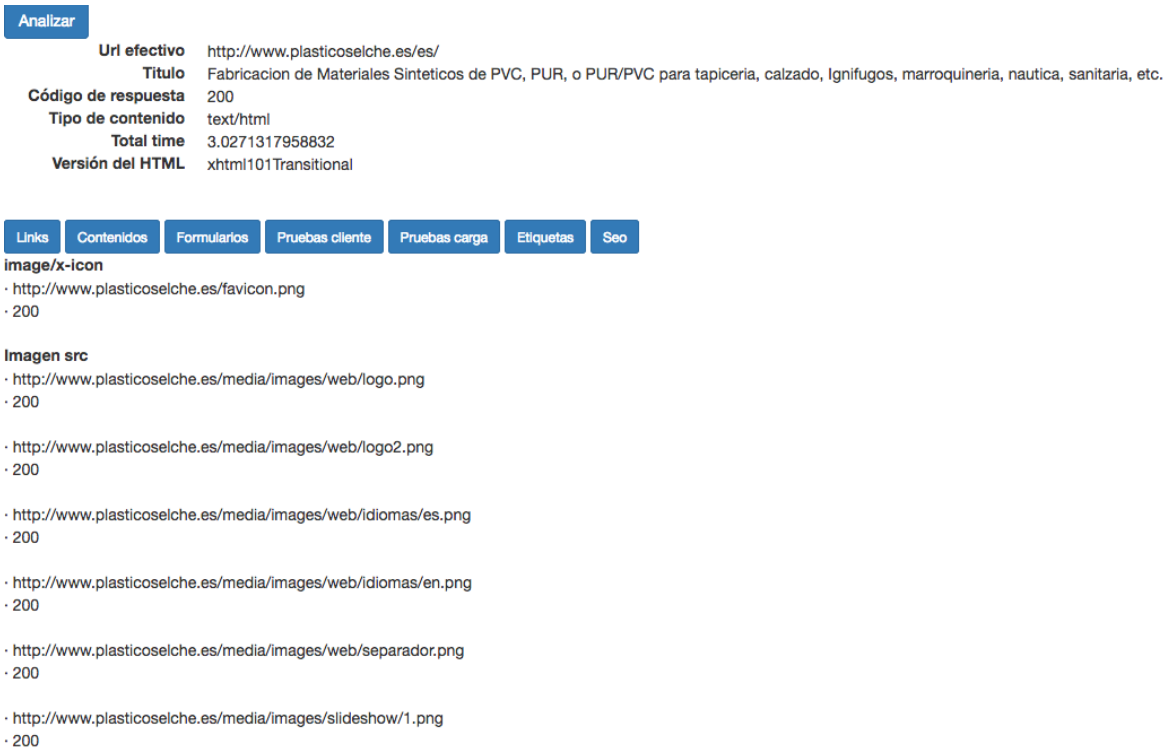
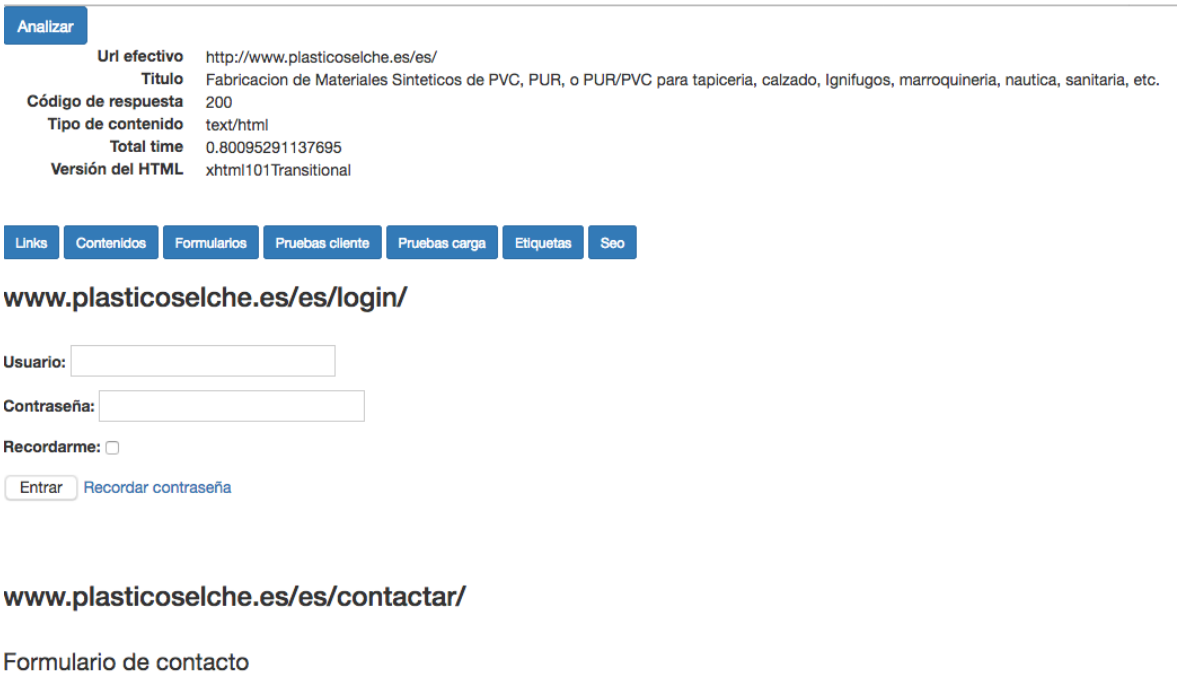


Figura 24: Contenidos

Formularios: La parte de Formularios representa los formularios obtenidos anteriormente. No son iguales debido a que se obtiene el estilo del de nuestra aplicación.



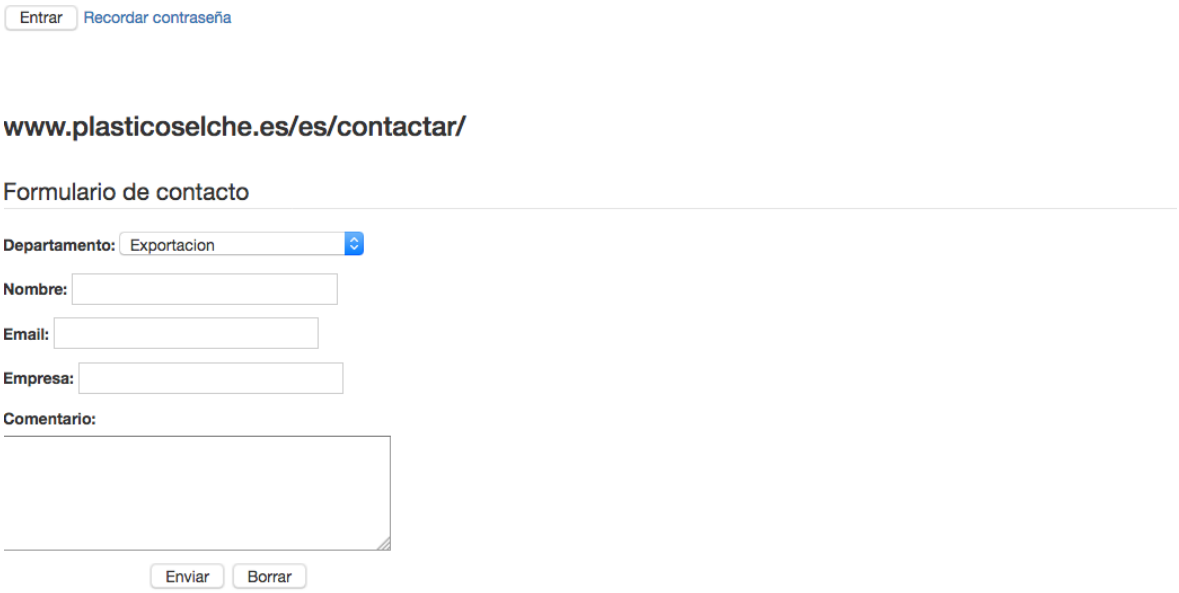


Figura 25: Formularios

Pruebas Cliente: El array de los formularios también se utiliza para el apartado de pruebas de cliente. Se pinta desde una clase PHP sin estilo e indicando el nombre de los campos, ya que tenemos que generar el contenido dinámicamente para poder utilizar los valores de la URL al enviar los datos que introduzcamos del desde teclado como usuarios del formulario.

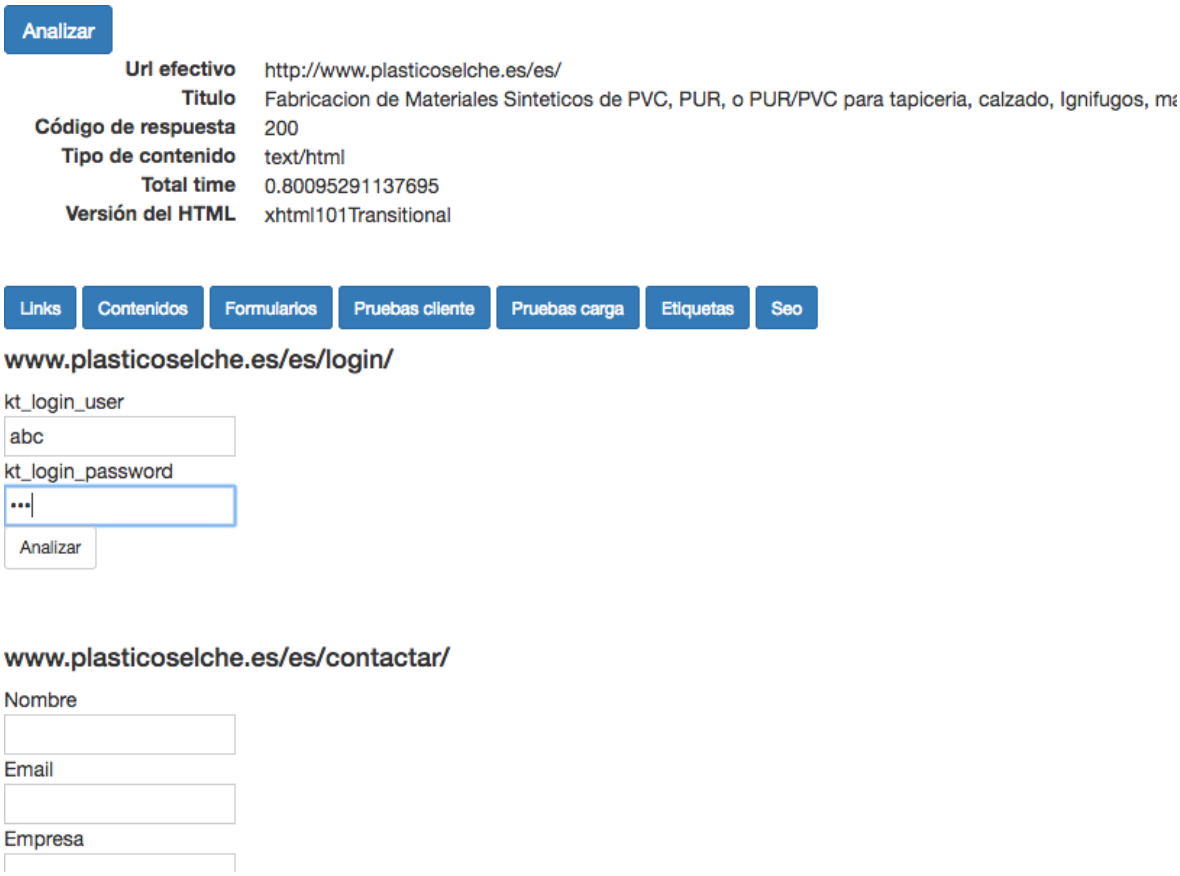


Figura 26: Pruebas Cliente 1

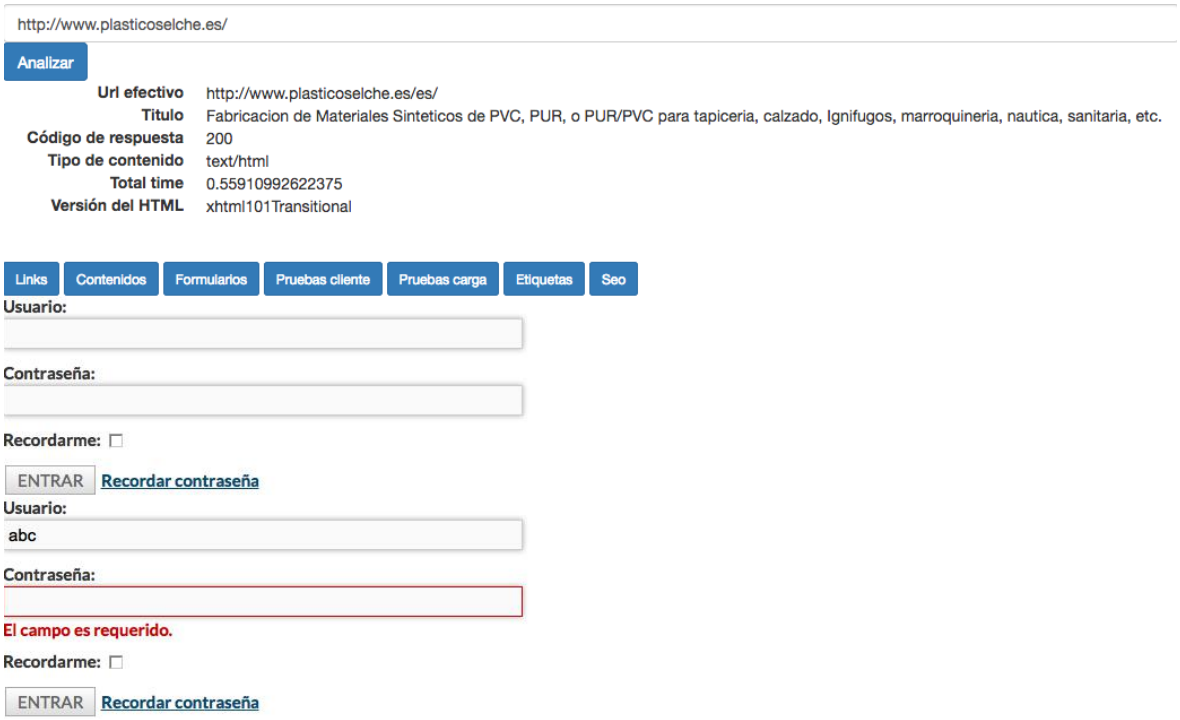


Figura 27: Pruebas Cliente 2

También guarda el valor del texto si se ha recibido un mensaje de alerta, cada vez más en desuso. Lo podemos ver resaltado en color azul bajo la imagen de después del envío

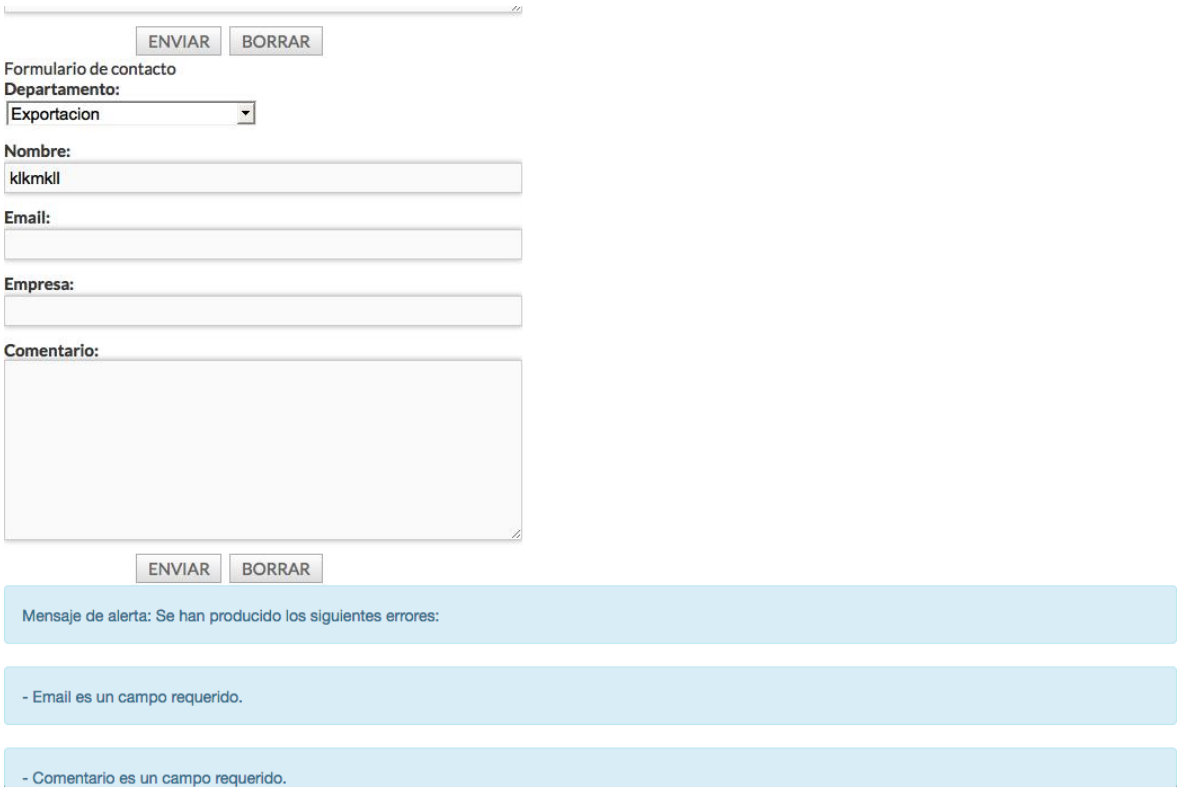


Figura 28: Pruebas Cliente 3

El script php obtiene estos archivos de texto y envía el contenido a la vista de nuevo para que represente las 2 imágenes en nuestra aplicación y los mensajes de alerta si se han producido.

Pruebas carga: Se lanzan pruebas de carga a la dirección del sitio web y a los formularios. Podemos indicarle el número de peticiones que queremos y número de peticiones de forma concurrente que se harán. Se harán peticiones a la dirección del sitio web y a la dirección de los formularios.

Se lanza el programa **Apache Bench** y se obtienen sus salidas, filtrando las que nos interesan

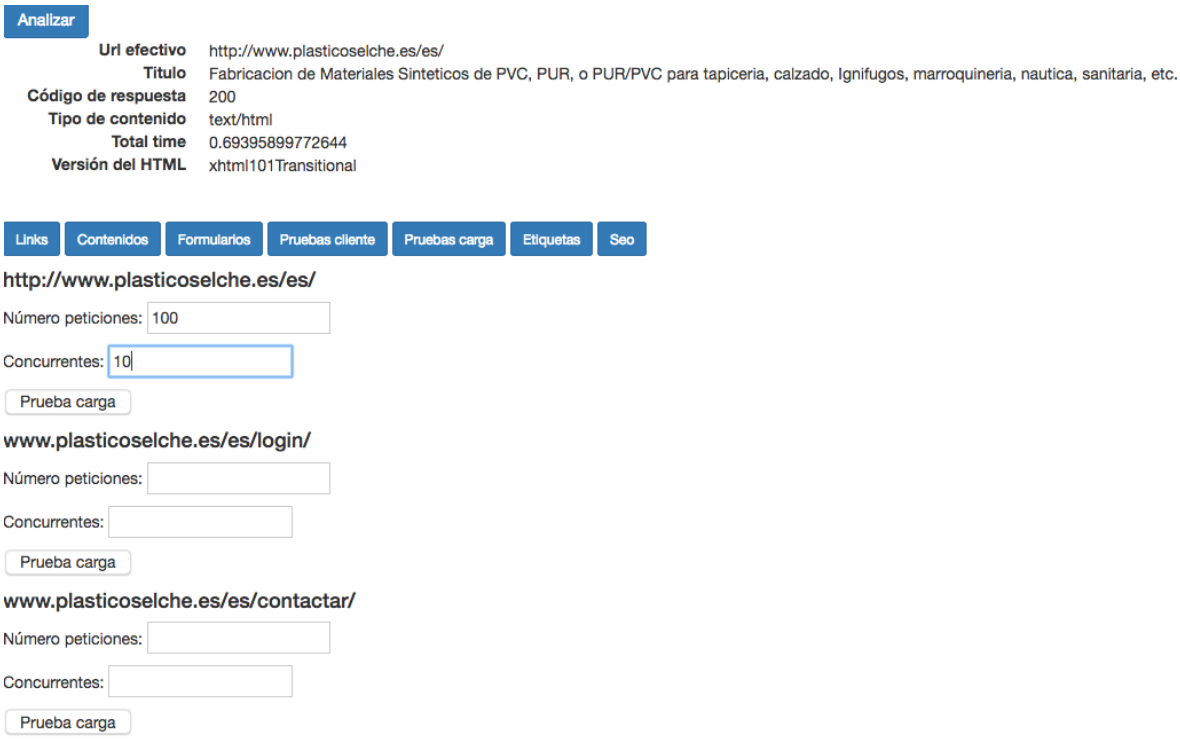


Figura 29: Pruebas carga 1

En los resultados podemos ver: el número de las peticiones por segundo que soporta el servidor, el tiempo en milisegundos por petición

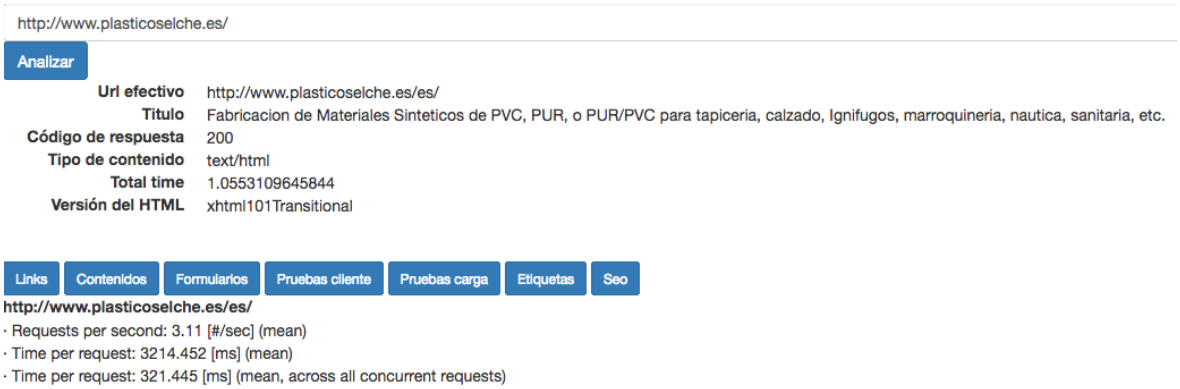


Figura 30: Pruebas carga 2

Etiquetas: En el apartado de etiquetas se muestra una gráfica `Highcharts` donde podemos ver el nombre de la etiqueta que aparece en la página principal del sitio web y la cantidad de veces que aparece en el HTML.

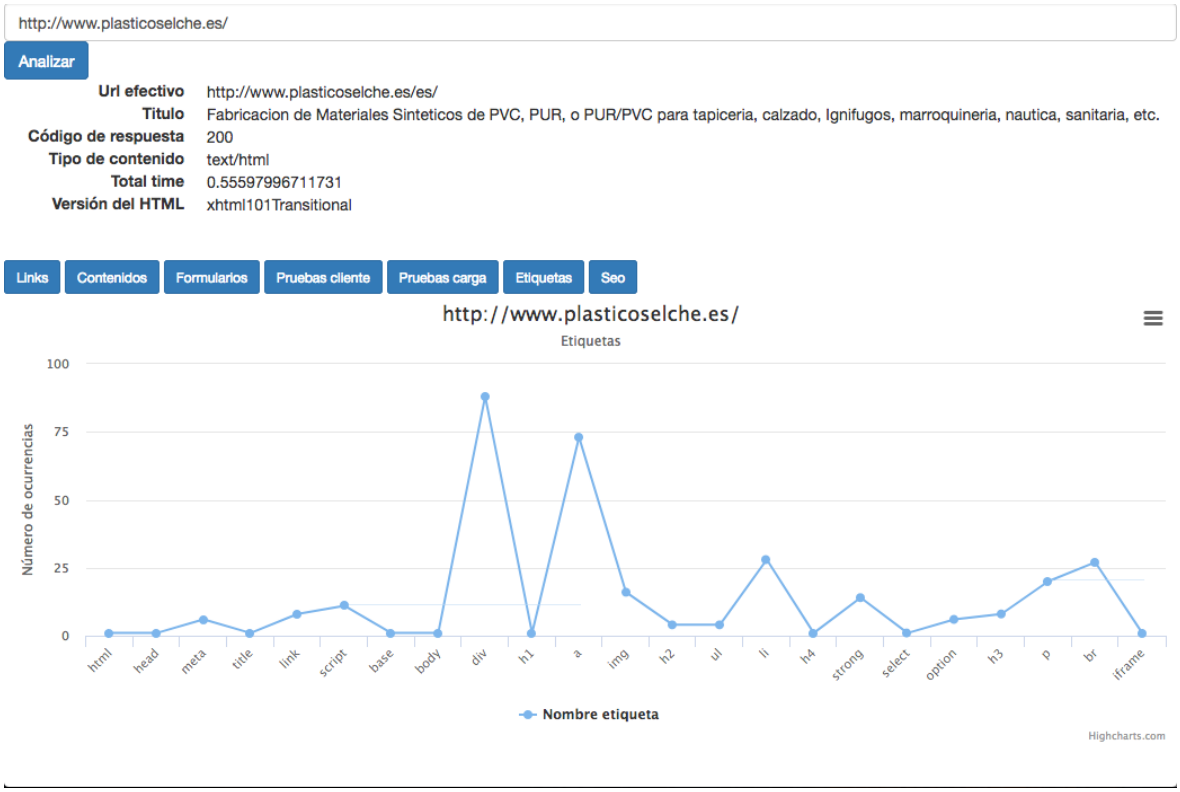


Figura 31: Etiquetas

Seo: En el apartado SEO donde se muestra el `<title>` para comprobar si es apropiado para el sitio web. Un poco lo mismo como la etiqueta `<h1>` y `<h2>`

Las etiquetas meta que contiene el sitio y sus respectivos valores. Como se ha visto antes son importantes para que los buscadores muestren el sitio web en sus resultados.

El valor del campo `alt` de las etiquetas `<img>`

Análisis Seo

Etiquetas básicas:

Titulo	Fabricacion de Materiales Sinteticos de PVC, PUR, o PUR/PVC para tapiceria, calzado, Ignifugos, marroquineria, nautica, sanitaria, etc.
Etiquetas h1	.
Etiquetas h2	Plásticos Elche **y** Comerplast
	Últimas **Noticias**

Etiquetas Meta:

	text/html; charset=utf-8
description	Plásticos Elche S.A.
keywords	Sinteticos, polipiel, eskai, piel sintetica, imitacion piel, PVC, PUR, mixtos, Ignifugos, tapiceria, sofas, marroquineria, carpetotecnia, contract, decoracion, antibacterias, transpirables, forros, etiquetas, termovirantes
author	mediaelx
viewport	width=device-width

Etiqueta alt de las imágenes

Url	Alt
http://www.plasticoselche.es/media/images/web/logo.png	Fabricacion de **Materiales Sintéticos** de PVC, PUR, o PUR/PVC
http://www.plasticoselche.es/media/images/web/logo2.png	Fabricacion de **Materiales Sintéticos** de PVC, PUR, o PUR/PVC
http://www.plasticoselche.es/media/images/web/idiomas/es.png	Español
http://www.plasticoselche.es/media/images/web/idiomas/en.png	English
http://www.plasticoselche.es/media/images/web/separador.png	.
http://www.plasticoselche.es/media/images/slideshow/1.png	Carpetotécnia

Figura 32: Seo 1

El contenido del archivo robots.txt si este sitio web lo contiene

http://www.plasticoselche.es/../imagenes/ferias/thumbnails/20_140x0.jpg	.
http://www.plasticoselche.es/media/images/web/logo-pie.png	Pesa

Archivo robots.txt

```
User-agent: *
Allow: /
Disallow: /*?
Disallow: /*.js$
Disallow: /*.css$
Disallow: /*.jpg$
Disallow: /*.html$
Disallow: /*.shtml$
Disallow: /*.php$
Disallow: /intramedianet
Disallow: /resources
Disallow: /css
Disallow: /js
Disallow: /Connections
Disallow: /includes
Disallow: /media
Disallow: /modulos
Disallow: /legal.php
Disallow: /lptd.php
```

Figura 33: Seo 2

6.3 Problemas encontrados

Durante el desarrollo de la aplicación se han encontrado los siguientes problemas.

Expresiones regulares: El primer problema fueron las expresiones regulares. Es un tema complejo. Aun así son una herramienta muy potente. Se puede hacer el tratamiento de cadenas mediante funciones viendo el índice del carácter etc. Cuando se encuentra la librería `simple dom parser` se solventa el problema ya que nos abstrae de todo esto.

Contenido a través de JQuery: Cuando se empezó a diseñar y programar el sitio web en sí, con tareas como la adaptación de datos para poder representar surgió el problema de la depuración. Si ejecutamos un script de php directamente desde el navegador, si hay errores se muestran directamente en la pantalla. Al hacer la petición AJAX con JQuery ya no. Aparecían pero en la pestaña de Red de Firebug, en el contenido de la petición POST.

PhantomPHP: Una herramienta que adapta la consola externa de `PhantomJS` en `Javascript` para usar en PHP de forma más o menos nativa. Con el requisito de tenerlo instalado en la máquina. Al intentar desarrollar una petición no funciona y los tutoriales de internet son escasos debido a la falta de comunidad. Al final se decidió por ejecutar la consola de `NodeJS`, es decir `CasperJS` y pasar los datos de PHP a ésta y de ésta a PHP mediante ficheros en el disco duro.

Algunas webs a base de javascript: No es lo normal Pero hay algunas webs que prácticamente su HTML no tiene etiquetas de formularios. Son funciones en `Javascript` que devuelven los caracteres para imprimirlo cuando hacemos click. Se podrían haber analizado los valores también, pero es mucho más complejo.

6.4 Implantación

Las siguientes aplicaciones y librerías son las que se han empleado en la solución final.

- Mozilla Firefox 54.0.1 Mac OS
    - Firebug 2.0.19
- Servidor Web Apache: Se ha utilizado XAMP 5.5.38-2 en el sistema operativo Mac OS
- Php versión: 5.5.38
    - Curl 7.45.0
    - DOM/XML 20031129, HTML support enabled
    - Regex Library
    - Json support 1.2.1
    - PCRE (Perl Compatible Regular Expressions) Support 8.38 2015-11-23
    - Shell bin/bash
- NodeJS 6.10.1
    - PhantomJS 2.1.1
    - CasperJS 1.1.4
- Apache Bench 2.3
- Bootstrap, JQuery, Highcharts: Scripts en archivos

En cuanto al hardware, el trabajo ha sido desarrollado en un portátil Macbook White Mid 2010:

- Procesador: Core 2 Duo
- Disco duro: SSD
- Memoria RAM: 8Gb

Un equipo modesto. Sin embargo si aumentamos la potencia, las funciones con coste lineal(1 foreach) y cuadrático(2 foreach anidados) Aumentará la velocidad de los scripts PHP.

Para ejecutar CasperJS y Apache Bench se han utilizado los ejecutables del equipo. También se podrían utilizar como servicios alojados en otro servidor web. La aplicación es web en cualquier proveedor de Hosting podría ponerse en producción, pues los requisitos no son muy elevados.

7 Conclusiones

Este trabajo de analizar sitios web en definitiva, me ha servido para aprender más a fondo cómo están hechos los sitios web. Se ha aprendido cómo están los enlaces en el HTML, con páginas HTML, scripts PHP o la ruta de la página HTML sin el servidor. También me ha llamado la atención la cantidad de enlaces de los sitios, desde una página del sitio web se puede acceder a todas las otras. Las respuestas del servidor cuando se hace un post, con un 200 casi todas las respuestas aunque se envíen datos erróneos o vacíos. La forma que tienen los sitios web mediante scripts en el lado del cliente para asegurarse que el usuario no envía datos erróneos. Algunos sitios no están optimizados para que se encuentren en el buscador, analizándolos con técnicas seo. En general la cantidad de formas que hay de escribir el HTML y que el navegador las procese todas.

Utilizar un lenguaje como PHP que no conocía apenas ha hecho que aprendiera un poco más. También está claro que tenemos una gran base de programación y no es una tarea muy complicada aprender un nuevo lenguaje, al final son los mismos conceptos.

Utilizar conceptos vistos en programación orientada a objetos y sistemas software, importantes para tener la arquitectura de la aplicación definida como modelo vista controlador.

Planificar el trabajo, establecer las características nuevas de las entregas y saber cuánto tiempo dedicarle a cada una también ha sido una tarea compleja.

Se han descubierto nuevas herramientas que no conocía potentes como CasperJS, para hacer pruebas funcionales de forma fácil. Además JQuery también me ha sorprendido por la facilidad para hacer peticiones AJAX en el sitio web.

Este trabajo podría extenderse más ya que se ha centrado en analizar sitios web básicos. Algunas posibles mejoras para este trabajo podían ser:

Tipos de entradas de formularios: Se podrían analizar más tipos de campos para las pruebas de formularios. Se han analizado los básicos son los de: input, password, select, textarea. Se pueden añadir algunos más como los de checkbutton, radiobutton etc…

Pruebas carga haciendo POST: En las pruebas de carga simplemente se hace la petición GET, una posible mejora es probar con el método POST y con más links del sitio web

Análisis seo a todos los links: Es conveniente analizar todas las páginas del sitio web, ya que las etiquetas generales del sitio pueden cambiar en cada una.

Análisis con usuario registrado: Cuando un usuario hace login tiene acceso a más características del sitio web.

Como conclusión en este trabajo de fin de grado he utilizado unos cuantos de los tantos conocimientos aprendidos en las asignaturas del grado. Me ha servido para ponerlos en práctica. También he aprendido otras realizándolo, que es de lo que se trata.

8 Bibliografía y referencias

Imagen de Mozilla Fundation:

- https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Funcionamiento HTML, HTTP:

- http://www.oreilly.com/openbook/webclient/ch03.html
- http://librosweb.es/libro/xhtml/capitulo_1/breve_historia_de_html.html
- http://web.ontuts.com/tutoriales/aprendiendo-a-utilizar-la-libreria-curl-en-php/

Versiones HTML:

- http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=444:icuales-son-las-versiones-de-html-diferencias-entre-html-4-y-html-5-significado-de-strict-cu00706b&catid=69&Itemid=192

Librería Simple Dom Parser:

- http://simplehtmldom.sourceforge.net/

Expresiones regulares:

- https://blyx.com/public/docs/expresiones_regulares_perl.html
- http://www.mclibre.org/consultar/php/lecciones/php_expresiones_regulares.html
- http://php.net/manual/es/reference.pcre.pattern.syntax.php

Pruebas cliente:

- http://casperjs.org
- http://www.davidam.com/docu/casperjs.html

Pruebas de Carga

- https://davidburgos.blog/como-hacer-pruebas-de-estres-tu-servidor/
- https://blog.diacode.com/testeando-el-rendimiento-de-tu-aplicacion-con-apache-bench

Selectores CSS

- https://www.w3schools.com/cssref/css_selectors.asp
- https://code.tutsplus.com/es/tutorials/the-30-css-selectors-you-must-memorize--net-16048

Seo

- https://es.wordpress.org/plugins/wordpress-seo/
- http://deteresa.com/archivo-robots-txt/
- https://support.google.com/webmasters/answer/6062608?hl=es

- https://support.google.com/webmasters/answer/79812?hl=es
- https://boluda.com/tutorial/guia-del-archivo-robots-txt/